

# Detecting translation errors in CAD surfaces and preparing geometries for mesh generation

*N. Anders Petersson & Kyle K. Chand*

This article was submitted to  
10'th International Meshing Roundtable, Newport Beach, CA, 10/7-10, 2001.

**August 27, 2001**

Approved for public release; further dissemination unlimited

## DISCLAIMER

This document was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor the University of California nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or the University of California, and shall not be used for advertising or product endorsement purposes.

This is a preprint of a paper intended for publication in a journal or proceedings. Since changes may be made before publication, this preprint is made available with the understanding that it will not be cited or reproduced without the permission of the author.

This report has been reproduced  
directly from the best available copy.

Available to DOE and DOE contractors from the  
Office of Scientific and Technical Information  
P.O. Box 62, Oak Ridge, TN 37831  
Prices available from (423) 576-8401  
<http://apollo.osti.gov/bridge/>

Available to the public from the  
National Technical Information Service  
U.S. Department of Commerce  
5285 Port Royal Rd.,  
Springfield, VA 22161  
<http://www.ntis.gov/>

OR

Lawrence Livermore National Laboratory  
Technical Information Department's Digital Library  
<http://www.llnl.gov/tid/Library.html>

# DETECTING TRANSLATION ERRORS IN CAD SURFACES AND PREPARING GEOMETRIES FOR MESH GENERATION<sup>1</sup>

N. Anders Petersson & Kyle K. Chand

*Center for Applied Scientific Computing  
Lawrence Livermore National Lab  
Livermore, CA 94551*

*August 27, 2001*

*UCRL-JC-144019*

## ABSTRACT

We have developed tools for the efficient preparation of CAD geometries for mesh generation. Geometries are read from IGES files and then maintained in a boundary-representation consisting of a patchwork of trimmed and untrimmed surfaces. Gross errors in the geometry can be identified and removed automatically while a user interface is provided for manipulating the geometry (such as correcting invalid trimming curves or removing unwanted details). Modifying the geometry by adding or deleting surfaces and/or sectioning it by arbitrary planes (eg symmetry planes) is also supported. These tools are used for robust and accurate geometry models for initial mesh generation and will be applied to in situ mesh generation requirements of moving and adaptive grid simulations.

**Keywords:** CAD models, CAD repair, Trimmed NURBS, IGES files

## 1. INTRODUCTION

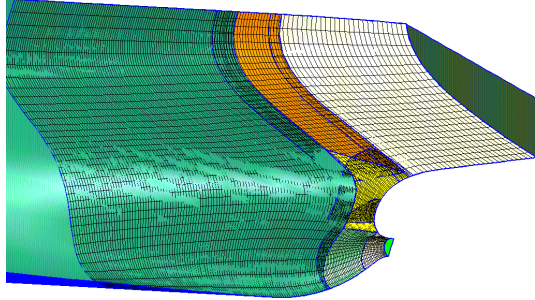
We present a suite of software tools and methodologies for preparing geometries for mesh generation. The geometries are obtained from IGES files translated from the internal representations of Computer Aided Design (CAD) packages. Preparation of these geometries includes identification and correction of translation errors as well as the cleansing of details regarded unnecessary for the intended analysis. A geometry is considered ready for mesh generation when it describes a watertight, consistent description of the modeled ob-

ject. Our approach consists of a collection of robust error detection utilities in combination with some automated correction facilities and an interactive interface for altering the geometry descriptions. This work describes one component of a larger set of grid generation tools in the **Over-**ture object-oriented framework [1].

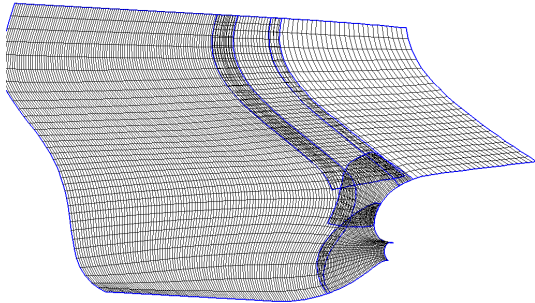
Our work supports the generation of overlapping meshes used to solve partial differential equations. Figure 1 depicts the overlapping surface grids over part of a ship hull. Generating these meshes requires fast and accurate evaluation of the underlying surface. If the underlying CAD representation contains errors, the mesh generator will not only reflect these errors, but in many cases will likely fail to generate meshes at all. While the tools are

---

<sup>1</sup>SUBMITTED TO 10<sup>TH</sup> INTERNATIONAL MESH-  
ING ROUNDTABLE, NEWPORT BEACH, CA, 2001.



(a) CAD surface with overlapping grids



(b) Overlapping grids

**Figure 1. CAD geometry and overlapping surface grids for a ship**

developed within the **Overture** overlapping grid framework, they are general enough to be useful for a variety of applications. Our intention is to make robust CAD models and efficient supporting software available for initial mesh generation as well as the in situ mesh generation for moving body and adaptive mesh simulations.

Researchers have proposed several approaches to preparing CAD models for grid generation. Some have suggested generic means to represent the geometry, sometimes interfacing directly with the

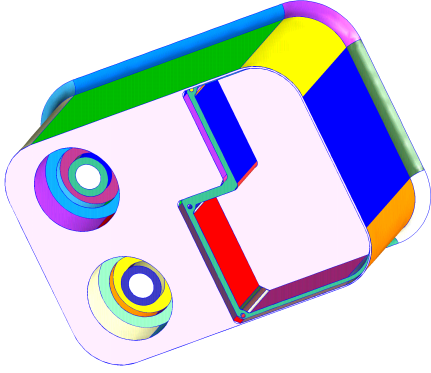
CAD packages themselves [2, 3]. Others attempt to deal with the representations provided by neutral file formats generated by CAD software [4, 5, 6]. The current work falls into the latter category since we desire to eliminate the dependence on proprietary software and to reduce memory requirements enabling the CAD model to be in memory during a moving or adaptive mesh refinement computation. While much work has been accomplished to automatically correct many problems encountered with these files [4, 5, 6], we still find many significant geometric deficiencies that require user input for resolution.

Our work is based on a flexible boundary-representation (B-REP) consisting of a network of surface patches, as described in Section 2. Currently, the surfaces are imported using the common IGES format. The plethora of errors found in these files is staggering. Errors span the range of simple problems such as slight curve mismatches to gross geometry errors such as missing surfaces or physically impossible geometry (such as zero thickness parts, duplicate surfaces, etc). Although we currently import models using IGES, the manipulations of the geometry are independent of the file format and we anticipate extending our software to also support the STEP format.

Correcting gross errors in the geometry begins with robust detection of the errors. In some cases, automated solutions are clear and reasonable to implement. Often, however, the geometry contains regions that make no sense whatsoever, leading to the need for a clean user interface for manual correction, as described in Section 3. Another need for user intervention arises when making meshes on a CAD model intended for production. Here models often contain parts irrelevant for the analysis and have more detail than is feasible to resolve on a mesh. Furthermore, large models can sometimes be divided along symmetry planes to decrease the overall size of the problem. These modifications imply substantial changes to the patchwork of surfaces, but can be handled in an efficient and nearly automatic manner by tools described in Section 4.

Error correction and model modification would be impossibly tedious without a clean and efficient user interface. The analyst must be told what the problems are and presented with an effective and efficient means to address them. Consequently, we have developed a graphical interface accessible through stand alone codes as well as the **Overture** library, which is described in Section 5.

During the final stage of geometry preparation, the topology is determined and a global trian-



**Figure 2. A Composite Surface consisting of many trimmed and untrimmed surfaces**

gulation is created on the corrected network of surfaces. Henshaw [7] has developed algorithms within **Overture** (similar to Steinbrenner et al’s [4]) to perform these steps. This new representation of the surface can be used to detect and correct any final gaps in the surface, and forms the basis of an efficient point projection scheme used in the mesh generator.

**Overture** is a C++ class-library consisting of tools for geometry manipulation, grid generation and solving partial differential equations on over-set grids [1, 7]. Our work will be made available through the next release of the **Overture** software, whose current release may be found at <http://www.llnl.gov/casc/Overture>.

## 2. GEOMETRY REPRESENTATION

Overture represents CAD geometries as a patchwork of surfaces, referred to as a composite surface. A composite surface consists of several component surfaces; component surfaces are represented with different shades in Figure 2. Each component surface is described as a mapping from the unit square in  $\mathbb{R}^2$  (the parameter plane) to the physical space in  $\mathbb{R}^3$ . Complex geometries often contain trimmed surfaces in their composite surface. Trimmed surfaces are comprised of parametric surfaces divided into “active” and “inactive” regions using curves in the parameter plane, as illustrated in Figure 3. Active pieces of the surface belong to the geometry description, while inactive regions should be ignored. Active and inactive regions are maintained by the orientations of the

trim curves in the parameter plane.

In addition to organizing the component surfaces into a patchwork, the composite surface provides an interface for computing the projection of a point in  $\mathbb{R}^3$  onto the closest point on the patchwork. This operation is central to generating a mesh on the composite surface, and is discussed in more detail by Henshaw [7].

Trim curves are given a parametric representation,  $\mathbf{r} = \mathbf{r}_c(t)$ ,  $0 \leq t \leq 1$ , where  $\mathbf{r} = (r, s)$  are coordinates in the parameter plane of the underlying surface. These curves are represented using Non-Uniform Rational B-Splines (NURBS) [8],

$$\mathbf{r}_c(t) = \frac{\sum_{i=1}^n N_{i,p}(t)w_i \mathbf{P}_i}{\sum_{i=1}^n N_{i,p}(t)w_i}.$$

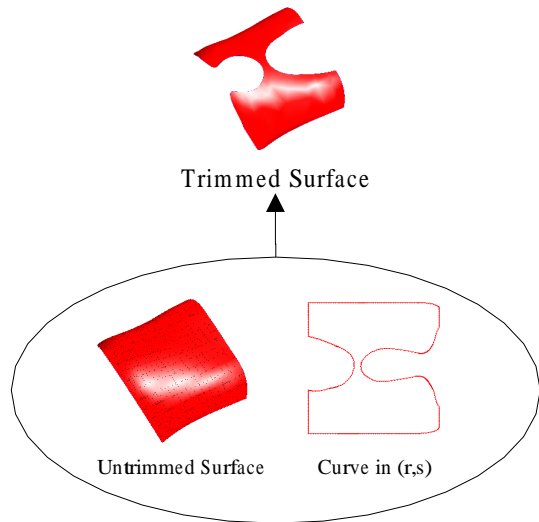
Here  $p$  is the degree of the spline,  $n$  the number of control points given by  $\mathbf{P}_i$  and  $w_i$  are the weights for each B-Spline basis function  $N_{i,p}(t)$ . Denote the knot vector by  $\mathbf{T} = (0, \dots, 0, T_{p+2}, \dots, T_{m-p-1}, 1, \dots, 1)^T$ , where  $T_i \leq T_{i+1}$  and  $m$  is the number of elements in  $\mathbf{T}$ . Note that the  $p+1$  leading zeros and trailing ones make the NURBS start at its first control point and end at the last. The basis functions are defined in a hierarchical manner,

$$N_{i,p}(t) = \frac{(t - T_i)N_{i,p-1}(t)}{T_{i+p-1} - T_i} + \frac{(T_{i+p} - t)N_{i+1,p-1}(t)}{T_{i+p} - T_{i+1}},$$

where the first function is given by

$$N_{i,1}(t) = \begin{cases} 1, & \text{if } T_i \leq t \leq T_{i+1}, \\ 0, & \text{otherwise.} \end{cases}$$

Each of the trim curves may be assembled by joining a number of curves into one continuous curve. When curves are joined, their degrees are first matched by elevating the lower degree curve. Then one curve’s knot vector is shifted and added to the other’s creating a knot vector  $\tilde{\mathbf{T}}$ , where  $0 \leq \tilde{T}_i \leq 2$ . This new knot vector is then scaled to lie in  $[0, 1]$  and the lists of control points are concatenated (removing any redundant points), see [8] for details.



**Figure 3. The trimming of a parametric surface using curves in the parameter plane**

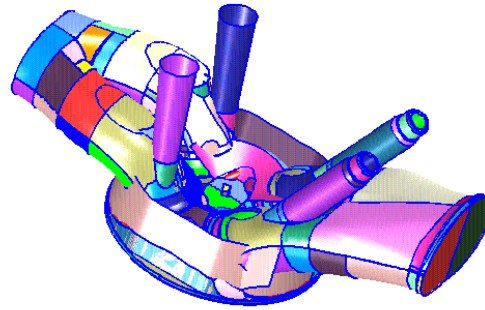
### 3. MENAGERIE OF IGES PROBLEMS

Geometry errors fall into two broad classes: errors in individual surface representations; and errors in the relationship between adjacent surfaces. Errors in individual trimmed surfaces are often due to trimming curves that do not fit together or intersect themselves. Incorrectly placed trim curves or surface boundaries lead to mismatches between adjacent surface components. As an example, consider the internal combustion engine geometry shown in Figure 4. Of 357 surfaces, 56 are improperly represented due to problems in the original IGES file. The majority of these problems were caused by incorrect trimming curves.

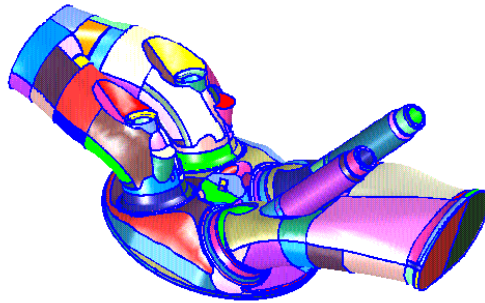
#### 3.1 Trim Curve Errors

Trim curve errors are detected automatically by examining each trimmed surface as it is read in or manipulated. Each trim curve must meet the following criteria:

- each curve must be continuous and periodic
- trim curves must be oriented properly to yield a bounded active surface
- the curve must remain within the unit square in the parameter plane
- the active region of the surface must not be extremely narrow



(a)

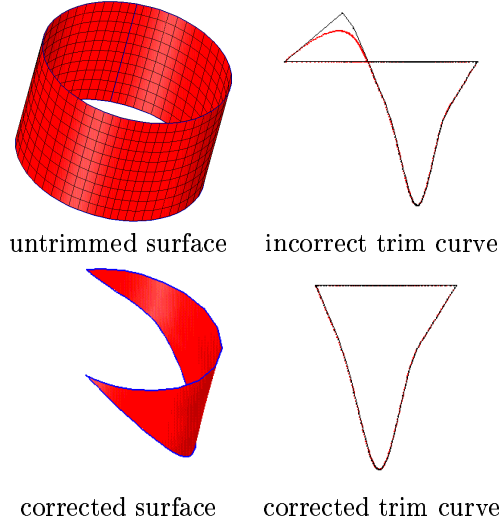


(b)

**Figure 4. Problematic Engine Geometry: (a) before correction; (b) after correction**

- a trim curve must not intersect itself

Checking the periodicity of a curve merely involves ensuring  $\mathbf{r}_c(0) = \mathbf{r}_c(1)$ . Trim curve orientation can be determined by computing and summing the signed areas of all the trim curves in parameter space. Counter-clockwise curves will have a positive area and clockwise curves negative. Since the active region of the surface must be bounded, there must be a counter-clockwise curve surrounding any clockwise one. Hence, the sum of all the signed areas must be positive. During the area computa-



**Figure 5. Self-intersecting NURBS trim curve: red curves are the trim curves, black their control points**

tion, the bounds of the curve in parameter space are computed to ensure that the curve lies within the unit square. Narrow surfaces can cause difficulties during mesh generation. We force the user to manually verify that any extremely narrow surfaces are indeed correct and necessary for the model. A curve is defined as extremely narrow if its area (in the parameter plane) divided by its arclength is less than some tolerance, say 0.001. Note that this absolute tolerance is not related to the size of the surface, since the parameter plane is always normalized to the unit square.

Curve self-intersection is a more difficult problem to treat efficiently. Figure 5 shows an example of a NURBS trimming curve that intersects itself. A general approach would discretize the curve into line segments and then perform a segment-segment intersection search. However, such a procedure would be very expensive since there can be thousands of trim curves in a complicated model. Instead, we use the polygonal approximation to the NURBS curve defined by its control points,  $\mathcal{P}^n = \{\mathbf{P}_1, \dots, \mathbf{P}_n\}$ . To make the technique more efficient, we first coarsen the polygon and perform an intersection check. Only if the coarsened polygon contains an intersection is the actual curve checked. We denote  $\mathcal{P}^{q-1} \subset \mathcal{P}^q$  as a coarse representation of  $\mathcal{P}^q$ . To form  $\mathcal{P}^r$ ,  $r < q$ , each point

$\mathbf{P}_i \in \mathcal{P}^q$ ,  $i = 2, 4, \dots, q-1$ , satisfying

$$\frac{(\mathbf{P}_{i+1} - \mathbf{P}_i) \cdot (\mathbf{P}_i - \mathbf{P}_{i-1})}{|\mathbf{P}_{i+1} - \mathbf{P}_i| |\mathbf{P}_i - \mathbf{P}_{i-1}|} > \cos(\alpha), \quad (1)$$

$$\frac{|\mathbf{P}_i - \mathbf{P}_{i-1}|}{|\mathbf{P}_{i-1} - \mathbf{P}_{i-2}|} < \delta, \quad (2)$$

$$\frac{|\mathbf{P}_{i+1} - \mathbf{P}_i|}{|\mathbf{P}_{i+2} - \mathbf{P}_{i+1}|} < \delta, \quad (3)$$

is removed. Since a trim curve is always periodic, the index should be interpreted modulus  $q$ , i.e.,  $\mathbf{P}_{-j} \equiv \mathbf{P}_{q-j}$  and  $\mathbf{P}_{q+j} \equiv \mathbf{P}_j$ . This procedure is repeated until no control points can be removed, or  $q \leq 4$ .

Criteria (1) ensures that points are only coarsened if the control polygon does not bend too sharply at  $\mathbf{P}_i$ . The latter two criteria attempt to keep neighboring polygon segments within a factor of  $\delta$  in length. Choosing  $\alpha = 110^\circ$  and  $\delta = 4$  appears to work well in practice, although we can not prove that the fine and coarse polygons have the same number of self-intersections. If an intersection is found in the coarse polygon, the intersection is verified for the original polygon  $\mathcal{P}^n$ . We remark that self-intersection of a NURBS control point polygon remains only a necessary condition for the curve to intersect itself. In cases where the actual curve has very narrow regions, it is possible for the control points to intersect but not the curve defined by the NURBS representation. However these cases remain rare and may be manually verified when encountered.

In IGES files, trim curves are often constructed from a number of sub-curves. Sub-curves are an additional source of errors, since they can not always be joined to form a continuous, periodic trim curve. We have also encountered cases when some sub-curves remain unused after the trim curve has been assembled. Such trim curves are regarded invalid, since the sub-curves are essential for the subsequent topology construction.

### 3.2 Surface Mismatch Errors

We rarely see catastrophic errors in the untrimmed surfaces, such as self-intersecting surfaces. More commonly, there are small gaps and overlaps between the surfaces. The current approach is tolerant to small mismatches since the algorithm for determining the topology adjusts trim curves to close gaps or overlaps, see [7]. However, large mismatches must be mended by hand.

## 4. GEOMETRY PREPARATION

### 4.1 Error Resolution

Correction of trim curve problems consists of first automatically detecting the reason a trim curve is invalid and then either fixing the problem, or helping the user to efficiently fix it. Automatic correction is available when there is enough information in the geometry to make the “correct” decision. For example, a curve can be forced to be periodic only if its endpoints lie within a certain tolerance of each other. Trim curves are usually manipulated in the parameter plane which avoids the difficulties of dealing with surface singularities and extremely thin patches. We have developed the following tools for the interactive modification of trim curves.

- Hide and show sub-curves. Hidden sub-curves are not considered during the assembly of the trim curve.
- Edit sub-curve (alter the control points and knots).
- Join endpoints with line segment. Joining the endpoints of two curves with a line segment is simply performed by creating a new NURBS curve representing a line between the points.
- Split a sub-curve. Splitting a NURBS curve is performed by adding control points and knots to ensure that the split curves coincide with the original curve, see [8] for details.
- Move a curve endpoint. Consider the case when the trailing endpoint is moved. If the new endpoint projects onto the parameter value  $t_e$ , where  $T_{m-p-1} < t_e$ , the last control point is simply moved to the new end point. If  $t_e \leq T_{m-p-1}$ , the curve is first split at  $t_e$  and the remaining part of the curve is discarded. The last control point is then moved to the new location of the end point.
- Truncating or extending two curves to their intersection. Curves are truncated or extended to their intersection using the endpoint move algorithm. The intersection is determined by either a Newton iteration if the curves intersect, or by the intersection of the tangents from the endpoints.
- Automatic assembly of sub-curves into a trim curve. Automatic assembly of the sub-curves into a continuous, periodic trim curve operates by attempting to join successive sub-curves until a closed loop has been formed.

- Manual assembly of sub-curves into a trim curve. An interactive “point-and-click” environment is provided for manual assembly of a trim curve.
- Creation and deletion of trim curves. New trim curves can be created by projecting edges of other surfaces onto the trimmed surface, or by a surface-surface intersection.

### 4.2 Patching Surfaces

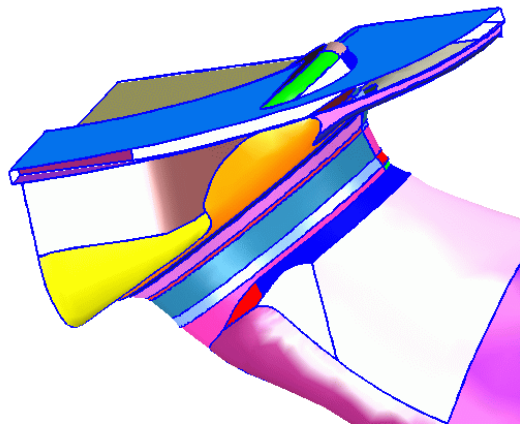
Gaps and overlaps between component surfaces that are too large to be corrected automatically (by the topology algorithm) need manual correction prior to mesh generation. Gaps can be handled by adding one or several surface patches. Figure 6 demonstrates this case by patching a gap in the engine geometry of Figure 4. The edges of the gap are selected by the user and a patch surface is automatically created using transfinite interpolation. To completely specify the surface when only two opposing edges are provided, straight lines are automatically created joining the opposing endpoints. Overlapping or intersecting surfaces can be repaired by projecting the boundary or intersection curves onto the surfaces involved and then adding these curves to the trimming.

### 4.3 Model Modification

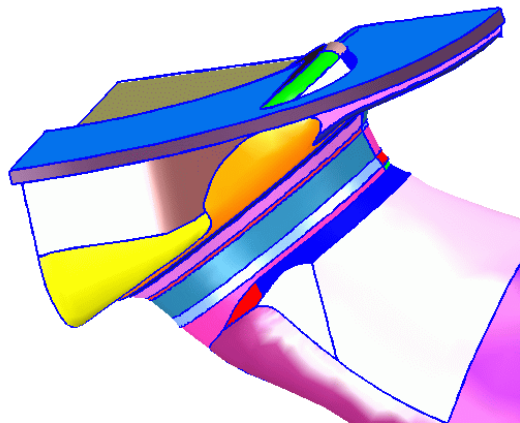
CAD geometries frequently contain far too much detail than is necessary or practical for a simulation. In the current work, unwanted details may be removed by deleting the surfaces that represent the unnecessary feature. Any holes that were part of the detail can be removed by deleting trim curves defining the hole, or by adding new surfaces covering the hole. For example, three small bolt holes from Figure 2 require removal. Figure 7(a) shows the three holes; most of the surfaces have been “hidden” to make the image more clear. Each bolt is represented by three surfaces and one trim curve in the outer surface. Once the surfaces are deleted from the model, each of the trim curves piercing the outer surface, Figure 7(c), can also be deleted resulting in the clean geometry of Figure 7(d).

In many cases a CAD model can be divided in order to take advantage of symmetries in the geometry or to perform an analysis on a specific section. We support arbitrary subdivision of the model using user defined cutting planes. Once a plane has been specified by the user, the intersection of the plane with the composite surface is automatically computed. Surfaces no longer in





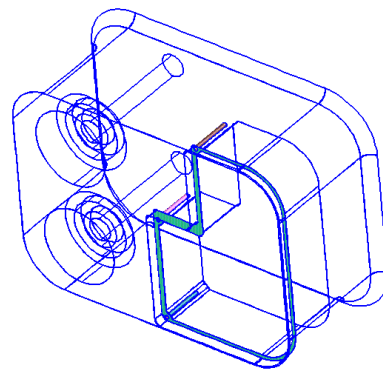
(a) surface with a missing patch



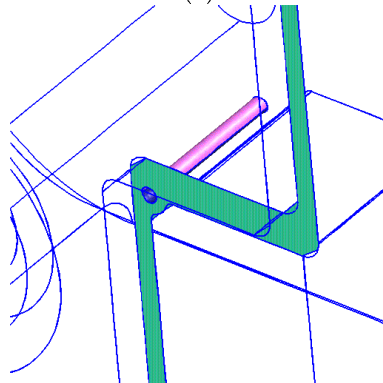
(b) surface corrected by adding a new patch

**Figure 6. Repairing a gap using a patch surface**

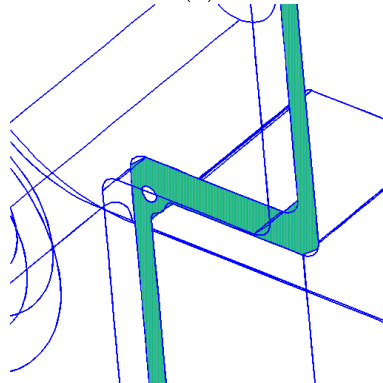
the model are removed while surfaces intersected by the cutting plane receive modifications to their trimming. Applying this technique to the geometry in Figure 4(b) provided a means for isolating the quarter of the model shown in Figure 8. Note also that the cylindrical feature, protruding from the geometry in Figure 4(b), has also been removed since it was not required for the intended simulation.



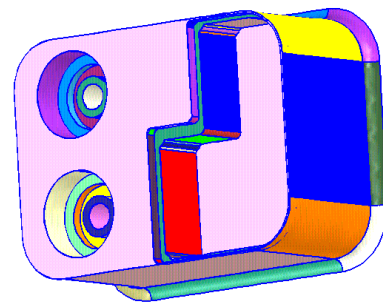
(a)



(b)



(c)



(d)

**Figure 7. Interactively Removing Unwanted Details**

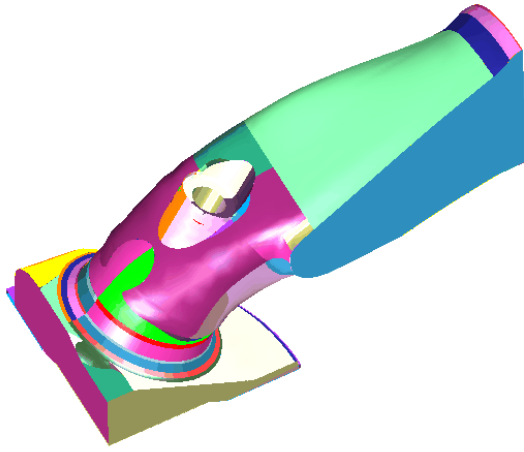


Figure 8. One quarter of the engine geometry

## 5. USER INTERFACE ISSUES

The graphical user interface should provide a clean and efficient interaction with the geometry. Ideally, an user interface makes any problem painfully obvious while providing a natural approach to the solution. Consider the invalid surfaces of Figure 4. Figure 9 depicts the broken surfaces alone (one operation in our interface), the valid surfaces have been hidden. Our software provides a three tiered interface for navigating, querying and manipulating these errors. The highest level interface presents basic functions dealing with the manipulation of the composite surface as a whole, see Figure 10(a). Here, features can be removed by deleting surfaces and trimming curves. Furthermore, surfaces and curves can be examined and edited by clicking on them. Various options are available for viewing the surfaces. Selecting a surface for examination and editing yields the interface in Figure 10(b). Here a trimmed surface can be viewed and edited in both physical and parameter spaces. The status of each trim curve can be obtained by selection of the appropriate curve. Finally, editing of a trim curve, perhaps the lowest level of manipulations, is accomplished using the interface in Figure 10(c). Emphasis has been placed on providing the user with a powerful set of manipulations backed by automatic error detection. For example, after editing a group of curves, one click is often enough to assemble the curves into a single trim curve, test the curve for its trimming suitability and then apply the trimming to the under-

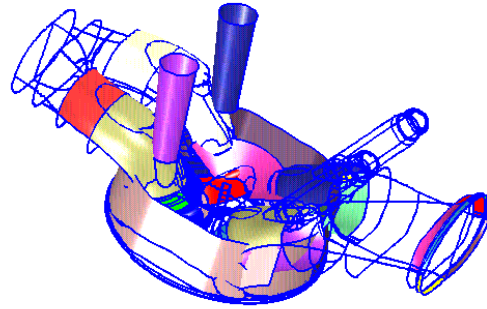


Figure 9. The broken surfaces of the engine geometry

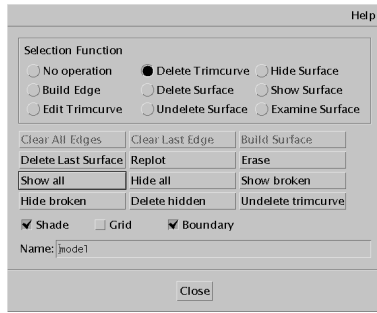
lying surface. If a mistake has been made, the system responds by telling the user what prevented the operation from occurring. Surfaces and curves can individually be queried for their properties, including the reason they are invalid.

Using this interface, we have significantly reduced the time needed to prepare geometries for mesh generation. For example, all errors in the IGES file for the engine geometry in Figure 4 can be corrected in less than half an hour (by an experienced user). The geometry is then ready for the topology algorithm, which is the last step before a mesh can be generated.

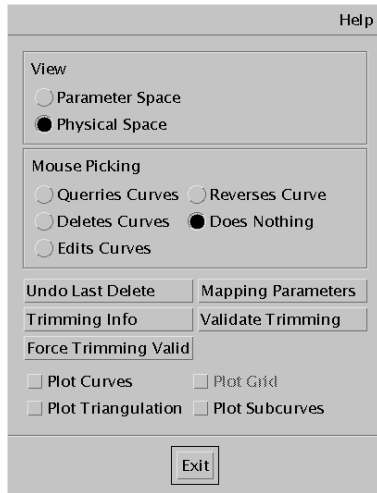
## 6. CONCLUSIONS

We have described techniques for correcting translation errors commonly occurring in IGES files. Methods for simplifying geometry for mesh generation were also discussed. The approach described here applies to geometry representations based on patched surface descriptions, and is not specific to the IGES format. The addition of a reader for STEP files is anticipated in the near future.

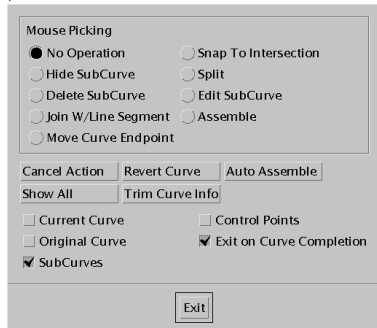
We envision first using the software interactively to prepare a CAD model for grid generation as well as generating the initial grid. The software can then be used as a geometry module and mesh generator during moving and adaptive grid computations, where repositioned or new grid points on the boundaries would be projected onto the corrected CAD geometry.



(a) Model editing interface



(b) Examining and editing a surface



(c) Editing a specific trim curve

**Figure 10.** Interfaces for editing the model, an individual surface and a trim curve

## 7. ACKNOWLEDGMENT

We thank Bill Henshaw and the **Overture** team for their advice and assistance. Olof Dahlberg of Volvo Cars is gratefully acknowledged for providing the engine geometry.

This work was performed under the auspices of the

U.S. Department of Energy by University of California Lawrence Livermore National Laboratory under contract No. W-7405-Eng-48.

## REFERENCES

- [1] David L. Brown, Kyle Chand, Petri Fast, Brian Gunney, William D. Henshaw, Brian Miller, N. Anders Petersson, Bobby Philip, and Dan Quinlan. *Overture Documentation, LLNL Overlapping Grid Project*, 2000. <http://www.llnl.gov/casc/Overture>.
- [2] Timothy J. Tautges. The common geometry module (CGM): A generic extensible geometry interface. In *Proceedings, 9th International Meshing Round Table*, 2000.
- [3] Silvio Merazzi, Edgar A. Gerteisen, and Andrey Mezentsev. A generic CAD-mesh interface. In *Proceedings, 9th International Meshing Round Table*, 2000.
- [4] John P. Stienbrenner, Nicholas J. Wyman, and John R. Chawner. Fast surface meshing on imperfect CAD models. In *Proceedings, 9th International Meshing Round Table*, 2000.
- [5] Geoffrey Butlin and Clive Stops. CAD data repair. In *Proceedings, 5th International Meshing Round Table*, 1996.
- [6] Andrey A. Mezentsev and Thomas Woehler. Methods and algorithms of automated CAD repair for incremental surface meshing. In *Proceedings, 8th International Meshing Round Table*, 1999.
- [7] William D. Henshaw. An algorithm for projecting points onto a patched CAD model. In *Proceedings, 10th International Meshing Round Table*. 2001.
- [8] Les Piegl and Wayne Tiller. *The NURBS Book*. Springer, 2nd edition, 1997.

